FIUSE: NVM-Free Secret Generation for Secure Chip-Scale Embedded Systems using SRAM PUF

Alfonso Cortés
alfonso.cortes@inria.fr
Inria, France

Sara Faour
sara.faour@inria.fr
Inria, France

Mališa Vučinić
malisa.vucinic@inria.fr
Inria, France

Filip Maksimovic
filip.maksimovic@inria.fr
Inria, France

Abstract

Low footprint embedded devices with single-chip microcontrollers, such as embedded RFID tags or single-chip systems, are becoming increasingly relevant for agricultural, implantable biomedical and supply chain tracking applications. The low footprint design requirement dictates which components can be integrated on the system. Relying on off-chip secure elements as a root of trust is, if practical, often not feasible or excessively expensive. Physically Unclonable Functions (PUFs) are commonly used to store or generate unique device secrets leveraging intrinsic nanometer-scale process variations. Existing reliable extraction methods still require Non-Volatile Memory (NVM) to store helper data, necessary to re-generate the secret from a noisy PUF response. These methods require either an off-chip memory or an expensive fabrication process that supports NVM integration. NVM-free PUF-based secret generation enables trusted computing on constrained embedded systems, while reducing fabrication costs. This paper proposes FIUSE, a standalone on-chip security peripheral that can be fabricated in common CMOS processes. This paper shows the feasibility of an NVM-free secret generator based on SRAM PUF, and gives insights on trade-offs between security, performance and resource utilization. This paper contributes with an open-source hardware implementation, from RTL to GDS, ready to be integrated with a RISC-V processor and manufactured in the IHP-SG13G process. FIUSE relies only on standard SRAM macros, which makes it portable to other technologies, and conventional cryptography functions can be easily added while remaining compatible with fabrication in regular CMOS processes.

CCS Concepts

• **Security and privacy → Hardware security implementation**; **Embedded systems security**.

1 Introduction

Chip-scale embedded systems, such as embedded RFID tags or single-chip microcontrollers, are relevant for applications that require minimum footprint devices, such as biomedical implantable devices. The principle is simple: integrating all functionalities within a single chip results in a compact device with no external components. The result is lower cost, as the system can be manufactured with only one ordinary CMOS process. Performing all functions within the same integrated circuit comes with many different challenges. For example, an integrated clock generator is very sensitive to ambient conditions such as temperature, compared to a conventional external crystal oscillator. Other components, including Non-Volatile Memory (NVM), can be directly integrated in the chip by using more sophisticated (and expensive) fabrication processes. The low-cost alternative is operating without persistent storage, which presents a new set of challenges.

The self-contained nature of chip-scale devices makes them inherently more resistant to many physical fault injection attacks [5], as well as side-channel attacks [19], because there are no exposed clock or data signals in the connection with external components. On the other hand, producing a Root-of-Trust (RoT) to enable security features is particularly hard without an external secure element and without NVM. Typically, a Root-of-Trust consists of features such as secret key generation, verification of trusted code and secure storage. All starts by deriving a root key from the first executable code and a Unique Device Secret (UDS) stored in secure NVM [30]. This root key is then used to derive multiple symmetric or asymmetric keys using Key Derivation Functions (KDF).

Many methods have been proposed to use Physically Unclonable Functions (PUFs) to reliably generate the Unique Device Secrets (UDS), thereby avoiding external modules or external key provisioning at manufacture time. SRAM cells, typically available in common CMOS processes, can be used as PUFs. That being said, most SRAM PUF secret extraction techniques still rely on NVM to store auxiliary information called *helper data*. PUF-based solutions can be used to re-generate keys or device secrets at each power-on as an alternative to NVM-based secure storage and key provisioning [29].

The goal of this paper is to enable security functions for chip-scale embedded systems using NVM-free UDS generation from SRAM PUF. The contributions are two-fold:

(1) We introduce Fully Isolated Unclonable Secret for Embedded (FIUSE), a self-contained hardware secret extractor optimized for implementation within a security peripheral;
(2) We evaluate the feasibility of secure NVM-free secret generation using SRAM PUF for UDS generation in chip-scale embedded systems.

The remaining of the paper is organized as follows. Sections 1.1 and 1.2 present the threat model and hardware requirements. Section 2 surveys conventional secret generation solutions and PUF-based methods. Section 3 provides background on NVM-free secret generation, as well as hardware support for RoT. Section 4 describes the design choices and implementation optimizations behind our proposed solution. Section 5 presents probabilistic analysis and

hardware implementation results revealing design trade-offs. Finally, Section 6 concludes this paper and presents for future work.

1.1 Threat Model

In the context of potential deployment scenarios for chip-scale systems, we consider an attacker that can:

- gain temporary or permanent physical access to deployed devices in the field,
- attempt to extract cryptographic keys or clone device identities,
- monitor side channels, such as electromagnetic radiation and power consumption,
- perform environmental stress attacks, such as inducing power glitches and manipulating the temperature,
- examine the device in a laboratory and attempt physical fault injections attacks, including the use of destructive equipment.

1.2 Requirements and Security Goals

We define hardware requirements for the design of a secret generation solution for chip-scale embedded systems:

R1: Single-chip. The solution must be fully integrated.
R2: Manufactureable in regular CMOS process. The solution can be fabricated in any conventional digital process, provided that SRAM macros are available.
R3: Minimum footprint. The hardware implementation is designed to reduce dedicated logic and volatile storage as much as possible.

In addition, we define the following security goals for the secret generation solution:

G1: Uniqueness of the device secret. Two devices will generate different secrets.
G2: Unclonability of the device secret. It is infeasible to create another physical device or model that reproduces the same secret.
G3: No persistent storage of the secret. The secret is stored in volatile memory and it is lost if the device is powered-off.
G4: Isolation from the software. The processor does not have access to the secret or the PUF. UDS and PUF exist in a self-contained environment, separated from the code execution.

2 State of the Art

Secure key storage in resource-constrained embedded devices traditionally relies on NVM such as EEPROM, Flash, or one-time programmable memory to store cryptographic secrets [33]. Commercial solutions such as Nordic Semiconductor's nRF series employ NVM-based key storage protected by access control mechanisms [24]. However, NVM-based approaches face challenges: *(1)* vulnerability to physical attacks including invasive probing, side-channel analysis, and fault injection [22], *(2)* high fabrication costs for integrating secure NVM in low-cost processes, and *(3)* limited write endurance and reliability degradation over time. Dedicated hardware security components such as external secure elements or on-chip isolated execution environments known as secure enclaves, provide robust protection. However, the additional silicon area, power consumption, and overall system cost make them impractical for ultra-low-cost single-chip systems [11].

PUFs offer an alternative paradigm by exploiting intrinsic manufacturing variations to generate UDSs when required, eliminating the need to store secrets in NVM [1, 8]. Silicon PUF implementations are based on diverse physical primitives: arbiter PUFs exploit race conditions in delay chains [23], ring oscillator PUFs measure frequency variations [28], butterfly PUFs leverage cross-coupled latches [15], and SRAM PUFs extract entropy from memory cell startup states [10].

2.1 SRAM Physically Unclonable Functions

Among PUF variants, SRAM PUFs offer compelling advantages for resource-constrained devices: *(1)* existing on-chip SRAM present in all fabrication processes [9, 10], *(2)* compatibility with standard memory compilers from vendors such as ARM, *(3)* high entropy density from the metastable startup behavior of cross-coupled inverters, and *(4)* ease of integration with existing microcontroller architectures [11, 26]. Large-scale field studies across hundreds of IoT devices have validated SRAM PUF reliability under diverse environmental conditions [11], demonstrating practical viability for real-world deployments. However, SRAM PUF responses exhibit inherent noise due to environmental variations (temperature, voltage) and aging effects (NBTI, HCI), necessitating error correction [20].

To extract reliable keys from noisy PUF responses, existing approaches combine Error-Correcting Codes (ECC) with helper data stored in NVM. Fuzzy extractors [3] formalize this paradigm: during enrollment, helper data is generated and stored; during reconstruction, it corrects errors in the noisy PUF response. Temporal Majority Voting [32] performs multiple readouts and applies majority logic to improve reliability, while Spatial Majority Voting [12] uses redundant SRAM cells. All these helper-based methods fundamentally require writable NVM to store helper data, limiting their applicability to NVM-free platforms [7, 14]. Alternative approaches improve native reliability either by modifying the SRAM circuit itself through custom cell designs with enhanced mismatch [17, 25] or controlled burn-in using hot carrier injection [16], Other approaches include leveraging stability-aware bit selection strategies based on analyses of aging effects and environmental variability [27], precise timing control [18], or customized power supply ramping [31]. While effective, these techniques require either custom cells incompatible with standard memory compilers, or depend on tightly controlled characterization and operating conditions that complicate integration into low-cost production flows.

2.2 NVM-Free Solutions

Recent work has explored helperless stabilization to eliminate NVM requirements. The probabilistic analysis by Bernardini *et al.* [2] evaluates trade-offs between average extraction reliability and yield of fabricated devices. On-Demand Helper Data (ODHD) schemes [4] generate helper data dynamically before each reconstruction by performing multiple PUF measurements, trading computation time for NVM elimination. However, ODHD was originally introduced

as an algorithmic concept without practical implementation or feasibility studies addressing resource constraints, execution time, or power consumption on real embedded platforms. This gap motivates our work: an NVM-free UDS generator based on SRAM PUF and compatible with standard fabrication processes, suitable for ultra-low-cost single-chip systems.

3 Background

3.1 SRAM Physically Unclonable Functions

A PUF is a circuit that maps an input to an output, where the map is unique to each fabricated device -or chip- instance. A PUF with no input is called a Physically Unclonable Constant (PUC). Every time an ideal PUC is queried it returns the same output, which can be used as an ID for the device, repeatable and unpredictable.

A typical SRAM cell, such as the 6-transistor circuit, can behave as a single-bit PUC, as it tends to initialize to a preferred state ('1' or '0') at power-on. The stability of a cell quantifies how consistently that fabricated instance initializes to its preferred state, i.e. how repeatable the query is. If we model the query on an SRAM cell binary PUC as the realization of a Bernoulli trial, the probability $P(X = 1) = p$ of it initializing to a '1' is different for every cell and depends mainly on nanometer-scale process variations that appear during fabrication. Assuming cells in an SRAM memory are independent variables, a set of cells is then a parameterized generator of binary PUCs where the parameter p of any cell is also the realization of a random variable [2]. An SRAM memory can also be seen as a PUF where the address is the input and repeatability depends on the probability for a sequence of cells of being stable. In addition, for the PUF to be unpredictable we care that the distribution of values of p is symmetrical so that the same number of cells prefer initializing to '1' as they do to '0' (uniformity). It is worth noting that depending on the technology, certain SRAM layouts result in PUCs that are not spatially independent.

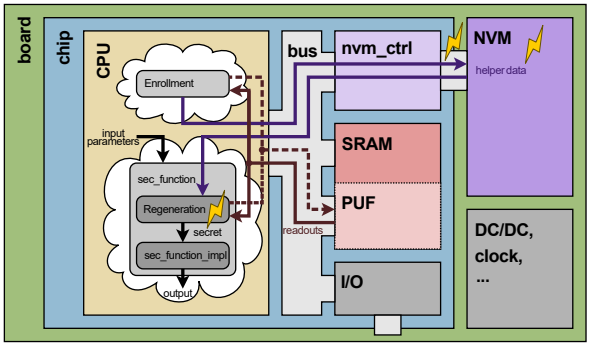
The raw output of a PUC (1 bit or more) can be stabilized by making multiple queries [2], which requires independence of consecutive queries. As shown in [18], the probability of a second readout being independent of the previous one increases with the turn-off time. In general, fuzzy extractors are used to ensure repeatability of the PUF output, by mitigating the effect of noise, and uniformity, by post processing the readouts.

It is important to note that the stability of a binary SRAM PUC is affected by ambient conditions such as temperature, supply voltage and electromagnetic noise, and can be altered by aging. The distribution of p for an SRAM on a certain process can be characterized experimentally for a set of ambient conditions as they did in [13].

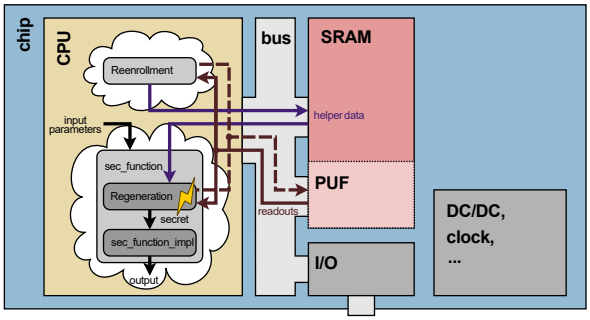
3.2 Secret Extraction

Fuzzy extractors are used to reliably generate a secret from SRAM PUF, despite the noisy readouts. During enrollment phase, an ECC encodes the SRAM output, using a large number of readouts and producing the secret, as well as helper data that is considered public information and must not leak the secret. During regeneration phase, the helper data allows to recover the secret from fewer readouts using the ECC. For example, in Threshold-based Majority Voting Scheme (TMVS) [6] the ECC parameters (codewords size and number, and threshold) determine how strict the selection of SRAM cell sequences will be during enrollment, governing a trade-off between regeneration reliability and required SRAM size.

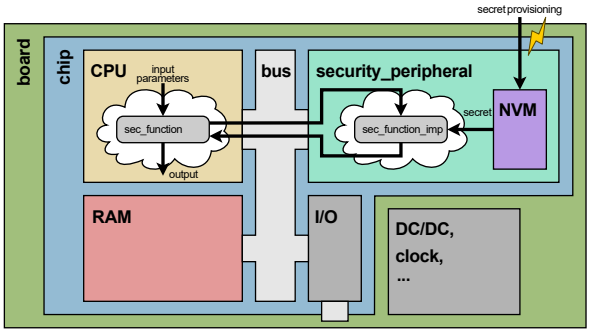
The ODHD extraction method is based on TMVS but is fundamentally different because it does not rely on persistent storage of the helper data in NVM. Helper data is lost every time the system is turned-off. Then, the enrollment phase is performed at each power-on in order to recover the same helper data from the PUF, instead of once in the lifetime of the device. The regeneration can be executed any time the secret is required. In ODHD, the secret reliability becomes a combination of (re)enrollment reliability and regeneration reliability.



(a) Software secret generation with PUF and external NVM.



(b) Software secret generation with PUF and without NVM.



(c) Secure secret storage with integrated NVM.

Figure 1: Block diagrams of different implementations of secret generation and storage for security operations.

3.3 Hardware Support for Root-of-Trust

In order to choose a secret generation method that complies with both the hardware requirements and the security goals, it is necessary to consider the microcontroller architecture and required hardware implementation. Figure 1 presents block diagrams of possible implementations of secret generation and storage using software or hardware, highlighting control signals (dotted arrows), data movements (solid arrows) and vulnerabilities (thunderbolts).

Figure 1a shows a conventional microcontroller board with on-chip SRAM and off-chip NVM, which does not comply with the hardware requirements. Other relevant circuits, such as power converters and oscillators, are also off-chip. Typical microcontrollers like this can perform a software secret generation scheme, e.g. TMVS. The connection to the external NVM enables attackers to tamper with the helper data, but this does not leak the secret.

Figure 1b is a block diagram of a single-chip microcontroller without NVM. Prototypes of such single-chip microcontrollers like [21] have shown great potential in terms of power consumption and footprint, which meets the hardware requirements from Section 1.2. This type of microcontroller can only perform an NVM-free software secret generation scheme, like ODHD, provided that the CPU is able to control the power of a bank of SRAM, which is common feature in off-the-shelf devices (i.e. power-gating). With ODHD, performing multiple readouts of the PUF is fundamental to increase reliability, at the expense of longer execution times.

In both Figure 1a and Figure 1b, insecure code can reproduce the secret extraction, unless the processor is designed with hardware mechanisms that ensure only the secure code can access PUF readouts. Furthermore, the generated secret exists at some point in the CPU, which is represents in itself an attack surface with potential vulnerabilities, failing security property **G4**.

Figure 1c shows another conventional microcontroller board that features a hardware security module with on-chip NVM, which requires additional expensive processes and does not comply with the hardware requirement **R2**. This kind of device does not perform secret generation, as the UDS is stored in the NVM during fabrication. This system can guarantee secure storage of the key during operation, but cannot prevent potential key exposure during the manufacturing process [29]. Furthermore, the key might not be unique for the device instance, failing **G1**, **G2** and **G3**.

4 FIUSE: Fully Isolated Unique Secret for Embedded

This section introduces Fully Isolated Unique Secret for Embedded (FIUSE), as a solution that enables secure chip-scale embedded systems while complying with the hardware requirements and security goals listed in Section 1.2. As shown in Figure 2, FIUSE is a security peripheral with a self-contained secret generation module. UDS generation is a primitive that enables any other security functions that is added to the peripheral and executed upon request from the code. The implementation of KDF or other security functions is out of the scope of this work.

4.1 Isolated Security Peripheral

The proposed solution is based on the chip-scale architecture shown in Figure 1b, with the addition of FIUSE, i.e. an isolated security

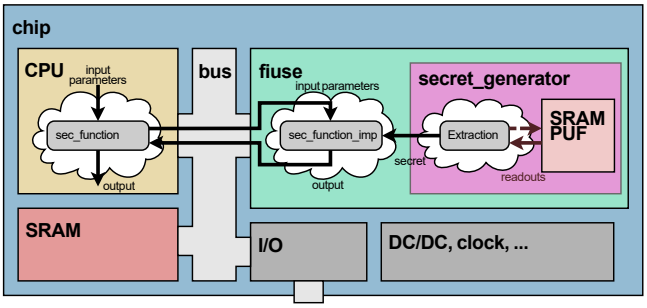


Figure 2: Proposed hardware secret generation, with dedicated SRAM PUF and without NVM.

module with a hardware implementation of a secret extractor based on SRAM PUF. With FIUSE, there is no software access to the SRAM PUF, which reduces the attack surface, and the system can still be fabricated in a common CMOS process.

The CPU only has read and write access to a set of registers in the interface of the peripheral, through the system bus, allowing the running code to provide operands, trigger the execution of security functions and fetch the returned values. The security functions can use the UDS generated by the FIUSE module from the SRAM PUF, as well as any secret keys derivated from it, but none of these are accessible from outside the peripheral. Compared to Figure 1b, the proposed system performs NVM-free secret extraction in hardware isolated from the software.

Since the secret generation is performed by an isolated hardware peripheral, at each power-on, the device performs the re-enrollment and the UDS can be stored in the logic inside of the peripheral. Consequently, there is no need to save helper data to perform on-demand regenerations and the re-enrollment BER becomes the only reliability metric of interest.

4.2 NVM-free Secret Extraction

To design a NVM-free secret generator for FIUSE, we start by considering the limitations of ODHD [4], an NVM-free fuzzy extractor that was introduced as an algorithmic concept to be performed in software. Compared to ODHD, our approach leverages the implementation as isolated hardware in the design of the algorithm.

The results in [4] suggest that smaller codewords and higher selection rates increase reenrollment reliability, to the detriment of regeneration reliability. Furthermore, in the absence of any persistent information of the PUF between extractions, there is no encoding of the cells that can improve the reenrollment reliability. Therefore, the proposed module simply iterates over the SRAM reading the number of bits required for the secret. There is no selection, nor helper data. Averaging multiple readouts is the only available mechanism to cope with PUF instability [2].

We present a design for the generator module, with a Register Transfer Level (RTL) implementation written in SystemVerilog[1]. The module is parameterized to choose the UDS size, the number of readouts and the size of the buffer. The buffer size governs a

[1] https://github.com/SDMote/SRAM_Key_Derivator/

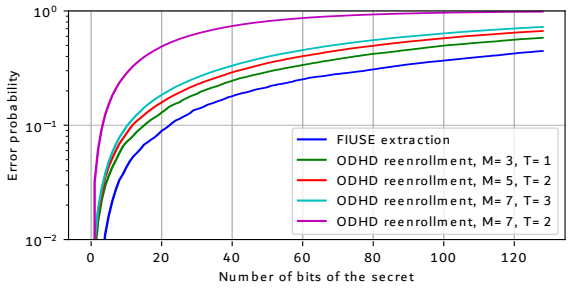


Figure 3: Extraction error probability versus secret size, comparing FIUSE and different configurations of ODHD.

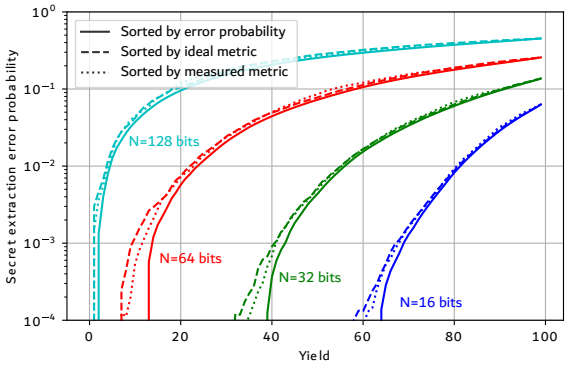


Figure 4: Extraction error probability versus yield of FIUSE instances for 81 readouts and different secret sizes.

trade-off between resource utilization and number of power-on cycles required to perform the readouts for all cells.

Since the stability of each PUF cell is determined at fabrication, we could theoretically identify devices with more unstable cells which would be more error prone during extraction. A stability metric is proposed to enable identification of poor performance device instances after fabrication, without leaking the secret.

The number of bit errors after R readouts of any bit i is

$$BE_i = \frac{R}{2} - \left| \sum_{j=1}^{R} r_{i,j} - \frac{R}{2} \right| \quad (1)$$

where $r_{i,j}$ is the value of readout j of bit i. Since any number of bit errors in the sequence produces an error in the generated secret, one very unstable cell in the sequence can be worst than many slightly unstable cells. Then, for a secret of N bits, we propose the following secret instability metric

$$SI = \sum_{i=1}^{N} 2^{BE_i} \quad (2)$$

Using powers of two helps reduce resource utilization. The number of significant bits in the Bit Errors is parameterized and governs a trade-off between precision of the metric and resource utilization.

5 Evaluation

This section provides probabilistic analysis of the extraction reliability using SRAM experimental data from the TSMC 65nm LP CMOS technology [13] and comparing to ODHD. Results of an RTL-to-GDS physical implementation are also shown, targeting fabrication with the IHP 130 *nm* SG13G2 open-source process.

5.1 Secret Reliability

The following simulations use data from 1000 readouts of 450000 SRAM cells to characterize the stability distribution of the PUF cells of the TSMC technology, in order to emulate any number of fabricated cell instances with the same distribution.

Figure 3 compares the expected reenrollment error of ODHD and the extraction error of FIUSE, for 81 readouts and a generated secret of up to 128 bits. Results are shown for different parameters of ODHD, where M is the number of bits of the codewords. $T = 0...E$ is the selection threshold, which is the radius of the hamming sphere around each codeword, where $E = \lfloor \frac{M-1}{2} \rfloor$ is the error-correction capability of the ECC. It can be seen that increasing M and reducing T reduces reenrollment reliability. Without persistent information of the PUF, mapping more than one cell to each UDS bit is detrimental for reenrollment. FIUSE, which corresponds to the trivial extraction from averaged readouts, outperforms any ODHD configuration.

5.2 Yield

Since not all fabricated devices have the same probability of extraction error, the average reliability can be improved by introducing a yield and discarding error-prone devices [2]. To sort the devices by reliability, we propose to measure the instability of the PUF cells using stability metric introduced in Subsection 4.2.

Figure 4 presents plots of error probability as a function of yield, when emulated PUF instances are sorted and kept or discarded depending on individual error probabilities or the proposed stability metric, for secrets of 16, 32, 64 and 128 bits. Sorting by the error probability of each instance achieves the minimum error probability, but in practice it cannot be measured directly without leaking the secret. In the *ideal* case, the stability metric is calculated with the exact stability of the emulated PUF cells and, in the *measured* case, the metric is estimated from 81 readouts. The *measured* metric sometimes outperforms the *ideal* metric because the metric does not correspond exactly to the error probability. However, the proposed metric with 81 readouts is already a good approximation for the purpose of identifying the most reliable devices.

It can be seen that choosing a yield can significantly reduce the error probability of a set of devices, particularly for a low number of secret bits. For example, choosing a yield of 60% reduces the error probability from 6.610^{-2} to 1.710^{-5} for a 16-bit secret.

5.3 Physical Implementation

The 256x8 single port SRAM macro, the smallest available in the SG13G2 130 *nm* process, is enough for an UDS of up to 2048 bits, with an area of 17547 μm^2. Table 1 presents the resource utilization of an RTL-to-GDS implementation in this technology, with a fixed secret size (N) and precision of the stability metric, and a target clock frequency of 25 MHz. Increasing the number of readouts (R), to improve reliability, results in higher resource utilization.

The size of the buffer (B) has as greater impact in the resource utilization and affects the number of SRAM power cycles, which dominates the execution time and is given by $R \cdot \lceil N/B \rceil$.

Table 1: Resource utilization of a 128-bit UDS generator for different number of readouts and buffer sizes.

Readouts	Buffer size	Flip-Flops	Std. cells area	Total area
-	*bits*	-	μm^2	μm^2
127	16	458	63547	81094
127	32	570	95940	113487
255	16	476	68100	85647
255	32	604	105379	122926

6 Conclusion

The proposed peripheral complies with the security properties and hardware requirements listed in Section 1.2, enabling secure computing in chip-scale embedded systems fabricated using a regular CMOS process. It has been proven that a trivial extractor outperforms ECCs in the absence of NVM, and increasing the number of readouts remains the only mechanism for the secret generator to reduce error probability. However, the reliability of the secret generator depends highly on the stability of individual SRAM cells of the technology. With the experimental data from [13], the error probability is $0,45$ for the generation of a 128-bit secret with 81 readouts and without yield. The proposed stability metric can be used to determine which fabricated devices have reliable UDS generation, enabling selection of the best performing ones.

Future work will involve the use of modified SRAM cells and in-memory computing to improve the probability of successful read-outs and the yield of fabricated devices.

References

[1] Abdulaziz Al-Meer and Saif Al-Kuwari. 2023. Physical Unclonable Functions (PUF) for IoT Devices. *Comput. Surveys* 55, 8 (2023), 1–31.

[2] Riccardo Bernardini and Roberto Rinaldo. 2016. Theoretical Limits of Helperless Stabilizers for Physically Unclonable Constants. *IEEE Transactions on Emerging Topics in Computing* 4, 1 (2016), 73–87. doi:10.1109/TETC.2014.2386137

[3] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. 2004. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. In *Advances in Cryptology – EUROCRYPT 2004 (Lecture Notes in Computer Science, Vol. 3027)*. Springer, 523–540.

[4] Sara Faour, Filip Maksimovic, Thomas Watteyne, Kristofer Pister, and Mališa Vučinić. 2026. ODHD: On-Demand Helper Data generation for reliable NVM-free key derivation from SRAM PUF. *Elsevier Computers & Security* 170 (2026), 104989. doi:10.1016/j.cose.2026.104989

[5] Sara Faour, Mališa Vučinić, Filip Maksimovic, David Burnett, Paul Muhlethaler, Thomas Watteyne, and Kristofer Pister. 2023. Implications of Physical Fault Injections on Single Chip Motes. In *2023 IEEE 9th World Forum on Internet of Things (WF-IoT)*. 1–6. doi:10.1109/WF-IoT58464.2023.10539380

[6] Sara Faour, Mališa Vučinić, Filip Maksimovic, David C. Burnett, Paul Muhlethaler, Thomas Watteyne, and Kristofer Pister. 2024. TMVS: Threshold-based Majority Voting Scheme for Robust SRAM PUFs. In *2024 IEEE Symposium on Computers and Communications (ISCC)*. 1–8. doi:10.1109/ISCC61673.2024.10733719

[7] Yansong Gao, Yang Su, Wei Yang, Shiping Chen, Surya Nepal, and Damith C. Ranasinghe. 2019. Building Secure SRAM PUF Key Generators on Resource Constrained Devices. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 912–917.

[8] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. 2002. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*. ACM, 148–160.

[9] Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. 2007. FPGA Intrinsic PUFs and Their Use for IP Protection. In *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. 63–80.

[10] Daniel E Holcomb, Wayne P Burleson, and Kevin Fu. 2008. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Comput.* 58, 9 (2008), 1198–1210.

[11] Peter Kietzmann, Thomas C. Schmidt, and Matthias Wählisch. 2023. PUF for the Commons: Enhancing Embedded Security on the OS Level. *IEEE Transactions on Dependable and Secure Computing* 20, 6 (2023), 5170–5184.

[12] Patrick Koeberl, Jiangtao Li, Anand Rajan, and Wei Wu. 2013. A Practical Device Authentication Scheme Using SRAM PUFs. In *Proceedings of the 5th International Conference on Trust and Trustworthy Computing (TRUST)*. 63–77.

[13] Blaz Korecic and Sara Faour. 2026. SCuM SRAM Characterization repository. https://github.com/bkorecic/scum-sram-evaluation/.

[14] Ashwija Reddy Korenda, Fatemeh Afghah, and Bertrand Cambou. 2019. A Proof of Concept SRAM-based Physically Unclonable Function (PUF) Key Generation Mechanism for IoT Devices. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–8.

[15] Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert-Jan Schrijen, and Pim Tuyls. 2008. The Butterfly PUF: Protecting IP on Every FPGA. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*. 67–70.

[16] Kunyang Liu, Xinpeng Chen, Hongliang Pu, and Hirofumi Shinohara. 2021. A 0.5-V Hybrid SRAM Physically Unclonable Function Using Hot Carrier Injection Burn-In for Stability Reinforcement. *IEEE Journal of Solid-State Circuits* 56, 7 (2021), 2177–2186.

[17] Kunyang Liu, Yue Min, Xuan Yang, and Hirofumi Shinohara. 2020. A 373-F^2 0.21%-Native-BER EE SRAM Physically Unclonable Function With 2-D Power-Gated Bit Cells and V_{SS} Bias-Based Dark-Bit Detection. *IEEE Journal of Solid-State Circuits* 55, 6 (2020), 1719–1729.

[18] Muqing Liu, Chen Zhou, Qianying Tang, Keshab K. Parhi, and Chris H. Kim. 2017. A data remanence based approach to generate 100% stable keys from an SRAM physical unclonable function. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. 1–6. doi:10.1109/ISLPED.2017.8009192

[19] Jacob N. Louie, Sara Faour, and David C. Burnett. 2024. Electromagnetic Side Channel Leakage Improvements Using Free-Running Oscillator Clock Reference. In *2024 IEEE Workshop on Crystal-Free/-Less Radio and System-Based Research for IoT (CrystalFreeIoT)*. 7–11. doi:10.1109/CrystalFreeIoT62484.2024.00006

[20] Roel Maes. 2013. *Physically Unclonable Functions: Constructions, Properties and Applications*. Springer.

[21] Filip Maksimovic, Brad Wheeler, David C. Burnett, Osama Khan, Sahar Mesri, Ioana Suciu, Lydia Lee, Alex Moreno, Arvind Sundararajan, Bob Zhou, Rachel Zoll, Andrew Ng, Tengfei Chang, Xavier Villajosana, Thomas Watteyne, Ali Niknejad, and Kristofer S. J. Pister. 2019. A Crystal-Free Single-Chip Micro Mote with Integrated 802.15.4 Compatible Transceiver, sub-mW BLE Compatible Beacon Transmitter, and Cortex M0. In *2019 Symposium on VLSI Circuits*. C88–C89. doi:10.23919/VLSIC.2019.8777971

[22] Francesca Meneghello, Matteo Calore, Daniel Zucchetto, Michele Polese, and Andrea Zanella. 2019. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. *IEEE Internet of Things Journal* 6, 5 (2019), 8182–8201. doi:10.1109/JIOT.2019.2935189

[23] Mohd Syafiq Mispan. 2018. *Towards Reliable and Secure Physical Unclonable Functions*. PhD thesis. University of Southampton.

[24] Nordic Semiconductor. 2024. *nRF Series: Security Features*. Nordic Semiconductor. https://www.nordicsemi.com/Products/Security

[25] Vinay C. Patil, Arunkumar Vijayakumar, Daniel Holcomb, and Sandip Kundu. 2017. Improving Reliability of Weak PUFs via Circuit Techniques to Enhance Mismatch. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 24–29.

[26] Manuel Penz, Martina Zeinzinger, Michael Kargl, Christian Steger, and René Mayrhofer. 2025. SRAM PUFs for Device Authentication on Resource-Constrained Systems. In *2025 12th International Conference on Communications Security and Privacy (CSP)*. 157–164.

[27] Andrés Santana-Andreo, Pablo Saraza-Canflanca, R Castro-Lopez, Elisenda Roca, and FV Fernandez. 2024. Reliability improvement of SRAM PUFs based on a detailed experimental study into the stochastic effects of aging. *AEU-International Journal of Electronics and Communications* 176 (2024), 155147.

[28] G. Edward Suh and Srinivas Devadas. 2007. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 9–14.

[29] Synopsys. 2024. *Flexible Key Provisioning with SRAM PUF*. Synopsys Inc. https://www.synopsys.com/designware-ip/security-ip/flexible-key-sram-puf.html

[30] Trusted Computing Group. 2024. *Hardware Requirements for a Device Identifier Composition Engine*. Trusted Computing Group. https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-a-Device-Identifier-Composition-Engine-Version-1.0-Revision-0.91_pub.pdf

[31] Wendong Wang, Adit Singh, Ujjwal Guin, and Abhijit Chatterjee. 2018. Exploiting power supply ramp rate for calibrating cell strength in SRAM PUFs. In *IEEE 19th Latin-American Test Symposium (LATS)*. IEEE, 1–6.

[32] Mengcheng Yue. 2024. A Two-Stage TMV SRAM PUF Preselection Method with Fewer ECC Resources. In *2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Vol. 7. 528–533. doi:10.1109/IAEAC59436.2024.10503621

[33] Shijun Zhao, Jiangnan Lin, Wei Li, and Bin Qi. 2021. Research on Root of Trust for Embedded Devices based on On-Chip Memory. In *2021 3rd International Conference on Computer Engineering and Application (ICCEA)*. IEEE, 464–469.